CTIA Certification

A Division of **CTIA**–The Wireless Association™

Wireless Device Low-Level API Specification

Revision 1.1
April 2005

CTIA Certification Program
1400 16th Street, NW, Suite 600
Washington, DC 20036

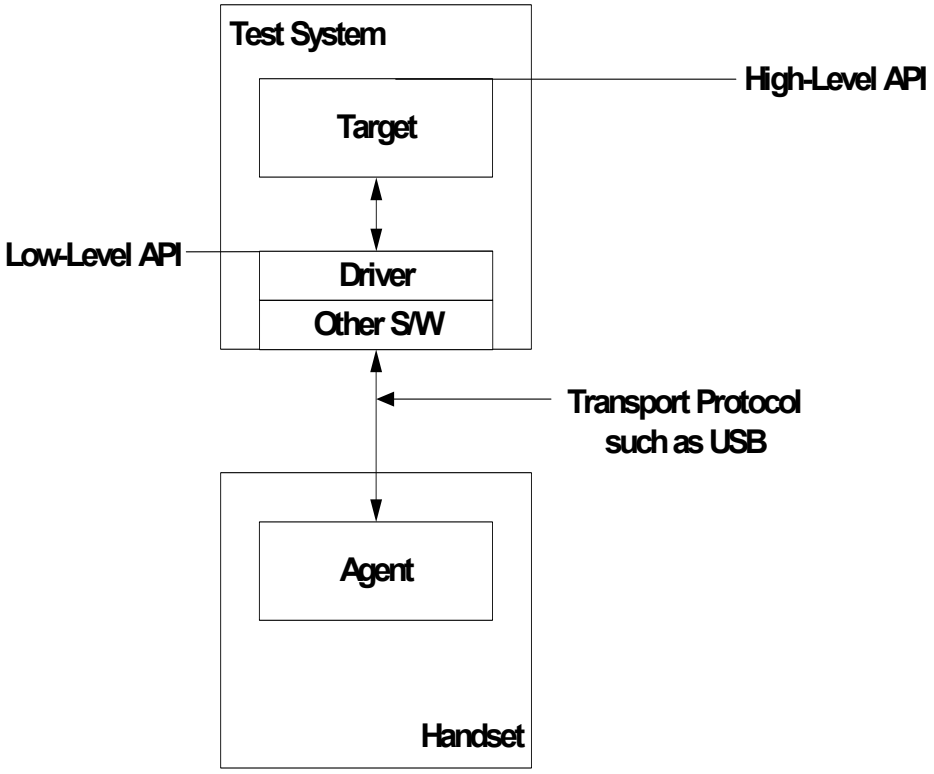
e-mail: certification@ctia.org
Telephone: 1.202.785.0081
www.ctia.org/certification

Table of Contents

1. Introduction

This document provides an interface definition for a low-level application-programming interface (API) between a mobile cellular phone (handset) and a test system or software program. This interface definition does not include transport protocol specification, leaving that particular implementation detail for designers. A separate document, Wireless Device High-Level API Specification, defines the high-level API. [1]

The general block diagram of this system is shown below:



In the drawing depicted above, the “Driver” and the “Other S/W” are optional pieces of software that may or may not be present, depending on the implementation of the capability by the handset manufacturer. If they are present, they tend to make the LLAPI and the HLAPI appear to be convergent, however the primary distinction that needs to be drawn between the two is that the HLAPI is independent of handset / manufacturer, while the LLAPI is dependent.

[1] http://www.ctia.org/certification/best_practices/index.cfm

2. Abbreviations, Acronyms, & Definitions

Agent	The capability residing on the handset that implements the API. This capability may exist in either software or hardware
API	Application Programming Interface
BLOB	Binary Large Object, a type of data buffer
HLAPI	High Level Application Programming Interface
LED	Light Emitting Diode
LLAPI	Low Level Application Programming Interface
PC	Personal Computer
RGB	Red, Green, Blue
Target	The software application residing outside the handset that communicates through the API with the Agent on the handset

3. Image Source Messages

3.1. Grab Image Message

The image source message requests an image of the device's display. The returned image is always the entire display area.

GrabImage request format:

Type	Description
UInt8	The Request Type, always a value of 1.

GrabImage successful response format:

Type	Description
UInt8	The Response Type, always a value of 4.
UInt16	The width of the returned image in pixels.
UInt16	The height of the returned image in pixels.
UInt32	A value that indicates the format of the BLOB. This is strictly a product of the contract between one target and its agent.
BLOB	The image.

Note that although the BLOB format is undefined, the information that is required at the Low Level API (LLAPI) is standard ".bmp" 24 bit color format, with the data for each pixel providing the RGB color value (one byte each). Dependent on the implementation method (driver or other software interface on PC side or not), the information can be compressed using loss less compression methods for transmittal between the handset and the driver / other software on the PC.

4. Keypad Messages

4.1. Key Press/Release Sequence Message

This message is used to press, wait and release keys on a keypad. It is a requirement of the target to ensure that the virtual key value is native to the agent. These device-dependent virtual key values are a product of the target-agent contract.

This is the method to use when more than one key at a time must be in the down state.

Key Press/Release Sequence request format:

Type	Description
UInt8	The Request Type, always a value of 8.
BLOB	A series of commands, giving the keys to be pressed/released or giving the time to wait before executing the next command. A byte value of 0x01 is followed by the virtual key value (UInt8) of the key to be pressed. A byte value of 0x02 is followed by the virtual key value (UInt8) of the key to be released. A byte value of 0x03 indicates that the following virtual key (UInt8) is to be pressed and then immediately released. A byte value of 0x04 if followed by the wait time in milliseconds (UInt32).

Key Press/Release Sequence successful response format:

Type	Description
UInt8	The Response Type, always a value of 22.

4.2. Key Sequence Message

This method is used to press and release a sequence of keys; this simulates a user typing on the keypad.

This method only supports pressing and then releasing one key at a time in sequence.

Key Sequence request format:

Type	Description
UInt8	The Request Type, always a value 25.
UInt32	The number of milliseconds to hold a key down before it is released.
UInt32	The number of milliseconds to wait between keys.
BLOB	The virtual key codes to press and release, each key is a UInt8

Key Sequence successful response format:

Type	Description
UInt8	The Response Type, always a value of 26.

5. Touch Screen Messages

Not all devices under test will have a touch screen. For those devices that don't, a standard error will be returned indicating that the hardware is not present.

The touch screen and the display are both considered to be laid out on an X-Y coordinate system with an origin of 0, 0 being the upper left corner of the **display**. This dictates that the touch pad coordinates are given in relation to the display. Therefore, pixel coordinate values can be positive or negative, with X increasing from left to right and Y increasing from top to bottom. A negative X value indicates a point to the left of the origin and a negative Y value indicates a point above the origin. If a touch screen does not have a display associated with it, as in the case of a laptop computer with a touch screen for controlling the pointer, the origin of 0, 0 is the upper left corner of the touch screen.

5.1. Tap Message

This message directs the agent to simulate a user tapping the stylus at one particular pixel.

Tap request format:

Type	Description
UInt8	The Request Type, always a value of 9.
Int16	The X-coordinate of where to tap the stylus, in pixels.
Int16	The Y-coordinate of where to tap the stylus, in pixels.
UInt16	The number of times to tap.
UInt32	The number of milliseconds to wait between stylus down and stylus up.
UInt32	The number of milliseconds to wait between stylus up and stylus down.

Tap successful response format:

Type	Description
UInt8	The Response Type, always a value of 17.

5.2. Stylus Down Message

This message requests that the agent simulate a user pressing down on a stylus at a specific pixel.

Stylus Down request format:

Type	Description
UInt8	The Request Type, always a value of 10.
Int16	The X-coordinate of where to move the stylus before pressing down, in pixels.
Int16	The Y-coordinate of where to move the stylus before pressing down, in pixels

Stylus Down successful response format:

Type	Description
UInt8	The Response Type, always a value of 18.

5.3. Stylus Up Message

This message requests that the agent simulate a user releasing the stylus at a specific pixel.

Stylus Up request format:

Type	Description
UInt8	The Request Type, always a value of 11.
Int16	The X-coordinate of where to move the stylus before releasing, in pixels.
Int16	The Y-coordinate of where to move the stylus before releasing, in pixels

Stylus Up successful response format:

Type	Description
UInt8	The Response Type, always a value of 19.

5.4. Move Stylus To Message

This request is used by the target to direct the agent to take the necessary action required to move a stylus to a given X, Y coordinate on the touch screen. The stylus can be in an up or down state.

Move Stylus To request format:

Type	Description
UInt8	The Request Type, always a value of 12.
Int16	The X-coordinate of where to move the stylus, in pixels.
Int16	The Y-coordinate of where to move the stylus, in pixels

Move Stylus To successful response format:

Type	Description
UInt8	The Response Type, always a value of 20.

6. Miscellaneous Messages

6.1. Ping Message

This is the equivalent asking the agent, "Are you still running?" It can also be used to determine round-trip latency. The only response possible is the successful response.

Ping returns a status that is only of real value when the agent is multi-threaded; It is not required that the agent's device have a multi-threaded OS.

Ping request format:

Type	Description
UInt8	The Request Type, always a value of 3.

Ping successful response format:

Type	Description
UInt8	The Response Type, always a value of 6.
UInt32	The Agent's status. Allowable values are: 0 = Agent is not busy. This is always the value for Agents where the OS does not support multi-threading because the Ping request cannot be processed until the previous request is completely handled. 1 = Agent is currently busy handling a previous request.

6.2. Batch Of Commands Message

This message type allows the target to batch up commands instead of sending them one at a time.

Because there is only one response from the agent to the target for this message, the commands in the batch cannot be commands that themselves request a reply. For example, *GrabImage* cannot be one of the commands in the batch because that message type gets data back in the response. Basically, only commands that are replied with a single-byte Response Type with no data are eligible to be contained in the batch.

Batch Of Commands request format:

Type	Description
UInt8	The Request Type, always a value of 27.
BLOB	The array of bytes that comprise the commands to be executed. The format of this BLOB is simply a sequence of the commands and their parameters.

Batch Of Commands successful response format:

Type	Description
UInt8	The Response Type, always a value of 21.

6.3. Sleep Message

This message tells the agent to sleep for some period of time. This message has no significance on its own but is defined only for its use in the *Batch Of Commands* message. This message is accepted by the agent if it is sent without being included in a batch but it really doesn't accomplish anything that can't be accomplished without doing anything.

Sleep request format:

Type	Description
UInt8	The Request Type, always a value of 28.
UInt32	The number of milliseconds to sleep before returning the response.

Sleep successful response format:

Type	Description
UInt8	The Response Type, always a value of 29.

6.4. Discrete State Message

This message requests state of devices associated with the device. Common devices that might be queried for state include LED's, ringer and vibrator.

Discrete State request format:

Type	Description
UInt8	The Request Type, always a value of 4.

Discrete State successful response format:

Type	Description
UInt8	The Response Type, always a value of 7.
UInt8	The number of discrete devices present on the device.
BLOB	The device name and state information. Format is Device Name / State.

7. System Messages

7.1. Get System Information Message

This request gets software version information and hardware device attributes.

Get System Information request format:

Type	Description
UInt8	The Request Type, always a value of 2.

Get System Information successful response format:

Type	Description
UInt8	The Response Type, always a value of 5.
UInt32	This field always contains a value of 1. The target uses this field to determine the endian-type from its format. Once the integer is extracted and placed into a C++ 32-bit *long*, if the value is 1 then the data is in little-endian (standard Windows) format. If the value is 0x01000000 then it is big-endian format. All further data extraction of multi-byte values will know the proper format.
UInt8	Major version number of agent's operating system.
UInt8	Minor version number of agent's operating system.
UInt16	Build number of agent's operating system.
UInt8	Major version number of the agent.
UInt8	Minor version number of the agent.
UInt16	Build number of the agent.
UInt16	The display's width in pixels.
UInt16	The display's height in pixels.
UInt32	The number of possible colors on the display.
UInt32	The size of the largest payload that the agent can accept from target.
UInt32	The size of the largest payload that the agent will send back to target.
UInt32	The size of the BLOB that follows; can be 0.
BLOB	This contains any target/agent specific information that the target and agent agree upon. When the preceding field is 0, this BLOB is not present.

APPENDIX A: Revision History

Revision	Date	Description of Changes
Rev 1.0	September 2004	• Initial Publication
Rev 1.1	April 2005	• Updated Introduction Section